Eurographics / ACM SIGGRAPH Symposium on Computer Animation (2008)
M. Gross and D. James (Editors)

Meshless Modeling of Deformable Shapes and their Motion

Bart Adams[1,2], Maks Ovsjanikov[1], Michael Wand[3], Hans-Peter Seidel[4], Leonidas J. Guibas[1]

[1] Stanford University
[2] Katholieke Universiteit Leuven
[3] Max Planck Center for Visual Computing and Communication
[4] Max Planck Institut Informatik

Abstract

We present a new framework for interactive shape deformation modeling and key frame interpolation based on a meshless finite element formulation. Starting from a coarse nodal sampling of an object's volume, we formulate rigidity and volume preservation constraints that are enforced to yield realistic shape deformations at interactive frame rates. Additionally, by specifying key frame poses of the deforming shape and optimizing the nodal displacements while targeting smooth interpolated motion, our algorithm extends to a motion planning framework for deformable objects. This allows reconstructing smooth and plausible deformable shape trajectories in the presence of possibly moving obstacles. The presented results illustrate that our framework can handle complex shapes at interactive rates and hence is a valuable tool for animators to realistically and efficiently model and interpolate deforming 3D shapes.

1. Introduction

Deformable shapes are used extensively in physics-based simulations for the animation of elastic and plastic solids. Although recent advances allow for interactive simulations, controlling the behavior of the deforming objects (for example by tweaking the physical forces) is difficult and the desired result is often only obtained after many iterations using trial and error. Instead of resorting to physics-based simulations, one could also explicitly model the poses of the deformable object at certain keyframes and interpolate a smooth motion in between (cf. Fig. 1). In many situations, such an approach accelerates the modeling process of an animated scene while quickly producing plausible behavior.

In this paper, we propose a shape modeling framework based on a meshless finite element formulation that allows efficient and realistic deformation modeling of complex shapes. From a coarse set of strategically placed sample points, called nodes, we compute a continuous deformation field that adequately represents the desired shape deformation. The optimized deformation field respects the original shape in the sense that it prefers locally rigid and volume preserving deformations.

Based on the shape modeling algorithm, we present a novel animation modeling framework that, given a few keyframe poses of the possibly deformed shape, computes a smooth and plausible trajectory that defines how the shape deforms and moves over a specified time interval. At every time instance, the shape is guaranteed to be realistically deformed

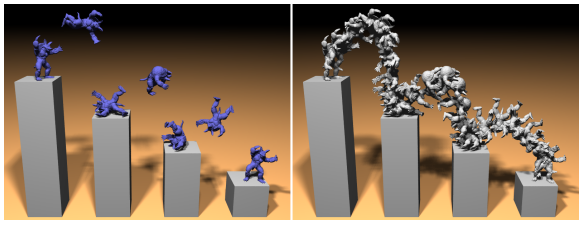

Figure 1: *Our method allows rapid modeling of deformable shapes and their motion. Left: keyframe poses obtained using our deformation algorithm. Right: smooth interpolated motion obtained using our shape interpolation algorithm.*

using the same constraints as above. Moreover, we show how collisions between the deforming shape and obstacles can be easily avoided. To maintain interactive modeling, we propose an adaptive temporal sampling strategy that keeps the number of unknowns low by only introducing computational frames at problematic time instances and interpolating a smooth motion in between.

Contributions We design a method for deforming highly complex shapes in real-time with guaranteed first-order accuracy of the interpolated deformation field. For this purpose, we extend existing work on meshless finite elements by introducing a novel coupling of nodes based on material distance. We then extend this deformation module to an animation modeling framework through highly efficient shape interpolation. By using our deformation field interpolation and only introducing frames in problematic areas, we greatly reduce the number of unknowns to be solved for. This al-

lows to create long and complex animation sequences. The key to efficiency is the decoupling of geometry and deformation and the use of specialized spatial and temporal sampling algorithms that introduce an adequate but low number of deformation handles in critical frames.

1.1. Related Work

This paper is related to a large body of work in geometric shape modeling, inverse kinematics, shape interpolation, motion planning, animation control, and meshless finite elements. We limit the discussion to the most relevant work in these areas.

Shape Modeling The core component of this paper is a deformation algorithm that allows to realistically deform a shape by specifying handle constraints. This problem has received considerable attention in recent literature. In the following we summarize the most closely related work and refer to [NMK*05, BS08] for an extensive survey on other deformation models.

A number of related methods perform shape deformations by direct mesh optimization such as [SK04, SYBF06, LSCO*04, SLCO*04]. The underlying motivation of these methods is to explicitly preserve the local shape properties while applying user-specified deformations. In a conceptually similar way, inverse-kinematics based methods restrict the space of natural deformations by either exploring the set of example poses [SZGP05] or by inferring the deformations on the skeleton structure of the shape [SZT*07]. Although mesh-based methods give a high degree of freedom in manipulating the shape, they suffer from the restrictive complexity of constraining and estimating per-vertex deformations. Multiresolution methods such as [BK03, SYBF06, BSS07] have been designed to improve efficiency.

Motivated by the intuition that in many scenarios, shape deformations can be encoded using only a few motion parameters, researchers have proposed techniques that use deformable models with a significantly reduced dimensionality as compared to the full geometric complexity (e.g., [JP02, JT05, DSP06, AFTCO07]). In this paper we use a reduced space deformation technique that allows computing long animation sequences for highly detailed deformable shapes. Of the large body of work in this area, the most immediately pertinent are [BPGK06] in which a prism based shell energy is formulated and solved efficiently, and [BPWG07] where a similar elastic energy is extended to rigid volumetric cells. Although the latter provides a simplified deformation field, it is both topology unaware and employs an interpolation scheme that results in solving a large sparse linear system making it prohibitively slow in our setting. Huang et al. [HSL*06] present a gradient domain mesh deformation technique that preserves volume and rigidity of limb segments of articulated figures. They propose a subspace technique by solving the problem on a control mesh. Somewhat differently, Funck et al. [vFTS06] design a set of vector field based deformation tools that guarantee non-intersecting and volume-preserving shape deformations. Their system seems to be more geared towards model creation as opposed to shape deformations. Our work is most similar in spirit to [SSP07] where the deformation field is discretized, solved for and interpolated using a sparse topology graph. Although we use a similar paradigm, we avoid estimating the rotation and translation components of the deformation field separately, and employ an interpolation scheme which guarantees first-order consistency. Moreover, our method introduces less unknowns for the same number of nodes. Finally, Stoll [Sto07] presents a tetrahedral deformation approach that iterates between a linear Laplacian step and a differential update step. Again, their deformation interpolation method is not guaranteed to be consistent.

Note that our deformation algorithm is also related to work that uses barycentric-like coordinates to interpolate the deformation field inside a coarse control mesh (see [FKR05, LKCOL07, JMD*07] for different flavors). Unlike these methods, however, we restrict the space of possible deformations to only include realistic, shape-preserving deformations. This facilitates the animator's task and allows intuitive shape modeling by only constraining or dragging points on the shape itself, without having to model and deform a cage.

Motion Modeling The second part of this paper is concerned with computing realistic deformable shape motions. Typically, these are obtained by controlling or modifying existing solutions obtained from a physics-based simulation algorithm (e.g., [WP95, PSE*00, PSE03] for rigid and [KKiA05, XZY*07] for deformable shapes). Such methods often require repeated simulations and adjustments using trial and error to obtain the required result. However, rather than modifying existing animations, our aim is to create an intuitive and interactive algorithm that computes the desired motions just from specifying a few keyframe poses. This problem was recently tackled by Hofer et al. [HPR04,Hof04] for rigid bodies by using curve design algorithms.

Finding motion paths for deformable objects is also related to morphing or shape interpolation where a smooth plausible deformation is computed interpolating two keyframe poses. A number of papers has considered this problem (e.g., [ACOL00, XZWB05, SK04, KMP07]). However, all these methods depend on the mesh or geometric complexity and are hence computationally expensive for long sequences involving detailed objects. Furthermore, none of them provide results for smooth deformable motion interpolation involving multiple keyframes in the presence of obstacles.

Meshless Finite Elements Our deformation representation is based on classical meshless finite elements (see [FM03] for a good overview), that were recently introduced in computer graphics for physically based animations (e.g., [MKN*04, PKA*05, GQ05, GP07]). We use a similar formulation as [MKN*04], but solve the inverse problem of computing the deformation field from given position con-

straints. Moreover, in the work of [MKN*04] nodes are coupled based on Euclidean distance, which does not allow adequate deformation modeling for nearby, but separated features (e.g. two adjacent fingers in a hand model). This problem was addressed by Pauly et al. [PKA*05] for the simulation of fracturing materials by using a visibility driven transparency criterion [OFTB96]. However, this method only allows separating parts cut by a crack surface and does not allow defining an appropriate nodal coupling for general shapes. To resolve these problems, we propose a novel algorithm that defines an adequate nodal coupling by using distances within the material that can be efficiently computed using a fast marching method [Set99]. This allows proper deformation modeling with the flexibility of traditional meshless algorithms such as easy sampling and a smooth and consistent deformation field representation.

We have used a similar deformation representation and optimization method in the context of reconstructing non-rigid shape and motion from 3D scanner data [WAO*08]. Apart from the different application, this paper presents an efficient GPU-based mesh deformation algorithm, improved and new motion modeling constraints, a different nodal sampling and coupling algorithm and, finally, a novel adaptive time discretization and interpolation scheme to reduce the computational cost.

1.2. Overview

We first describe the meshless shape function approximation theory in Sec. 2 that allows to reconstruct continuous functions with the desired order of consistency from values sampled at irregularly distributed points. It will form the basis of our shape deformation and motion planning framework. Next, in Sec. 3 we present our method to model realistic shape deformations. In Sec. 4, we add the time dimension and describe how the framework of Sec. 3 can serve as the basis for a motion planning algorithm for deformable objects. Sec. 5 gives details on the numerical solver. Sec. 6 concludes the paper with a discussion of the obtained results.

2. Meshless Shape Functions

Our deformation framework solves for the unknown deformation field at discrete points in space and time. In order to reconstruct a continuous deformation field, we use meshless shape functions that can be constructed to guarantee any order of desired consistency, independently of the underlying (possibly irregular) sample spacing. We briefly discuss the construction of these shape functions. For a more in-depth derivation and discussion, we refer to [FM03].

Given N discrete sample points $\mathbf{x}_i \in \mathbb{R}^d$ and associated function values $f_i \in \mathbb{R}$, we wish to retrieve the continuous function $f(\mathbf{x}) : \mathbb{R}^d \mapsto \mathbb{R}$ that approximates the function the samples are taken from. We additionally require the approximation to reconstruct functions up to order n exactly. As shown in [FM03], one can derive *shape functions* $\Phi_i(\mathbf{x}) : \mathbb{R}^d \mapsto \mathbb{R}$ associated with each sample point $\mathbf{x}_i$ that yield the desired approximation $f(\mathbf{x})$ of the form

$$f(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x}) f_i. \tag{1}$$

Given a complete polynomial basis $\mathbf{p}(\mathbf{x})$ of order n (for example for $d = 3$ and $n = 1$ we have $\mathbf{p}(\mathbf{x}) = [1\ x\ y\ z]^T$, with $\mathbf{x} = [x\ y\ z]^T$), the shape functions are defined as

$$\Phi_i(\mathbf{x}) = \mathbf{p}^T(\mathbf{x})[\mathbf{M}(\mathbf{x})]^{-1} w_i(\mathbf{x}) \mathbf{p}(\mathbf{x}_i). \tag{2}$$

Here we use the compactly supported weight functions $w_i(\mathbf{x}) = \max(0, (1 - d^2(\mathbf{x}, \mathbf{x}_i)/r_i^2)^3)$ that smoothly decay with increasing distance $d(\mathbf{x}, \mathbf{x}_i)$, where r_i is the support radius associated to the sample point $\mathbf{x}_i$. The matrix $\mathbf{M}(\mathbf{x})$ is called a *moment matrix* and is defined as

$$\mathbf{M}(\mathbf{x}) = \sum_{i=1}^{N} w_i(\mathbf{x}) \mathbf{p}(\mathbf{x}_i) \mathbf{p}^T(\mathbf{x}_i). \tag{3}$$

By using compactly supported weight functions, the summation in Eq. 1 can be limited to all sample points $\mathbf{x}_i$ s.t. $w_i(\mathbf{x}) > 0$. Note that the shape functions only depend on the discrete sample points $\mathbf{x}_i$ and not on the associated function values f_i. Hence, given a fixed sampling, the shape functions can be reused to approximate any function.

The derivative of Eq. 1 to the k-th component of $\mathbf{x}$, is obtained as

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_{(k)}} = \sum_{i=1}^{N} \frac{\partial \Phi_i(\mathbf{x})}{\partial \mathbf{x}_{(k)}} f_i, \tag{4}$$

where $\partial \Phi_i(\mathbf{x}) / \partial \mathbf{x}_{(k)}$ can be computed from Eq. 2 using the product rule and by using the fact that $\partial(\mathbf{M}^{-1}) / \partial \mathbf{x}_{(k)} = -\mathbf{M}^{-1}(\partial \mathbf{M} / \partial \mathbf{x}_{(k)})\mathbf{M}^{-1}$ with $\partial \mathbf{M} / \partial \mathbf{x}_{(k)}$ obtained from Eq. 3. As will be discussed below, Eq. 4 will be used to constrain the shape deformations to be locally rigid and volume preserving, to deform the shape's surface normals and to constrain the shape's velocity. Similarly, second-order derivatives can be computed [FM03] that will be used to constrain the shape's acceleration.

Note that Müller et al. [MKN*04] present a different method to compute derivatives. Their approach only allows computing derivatives at sample points $\mathbf{x}_i$, as opposed to Eq. 4 that allows evaluation at general positions $\mathbf{x}$. Moreover, their derivative approximation is only first-order accurate, while Eq. 4 presents an *exact* analytic formula for the derivatives of the function $f(\mathbf{x})$.

3. Shape Deformations

In this section we propose an efficient representation and optimization strategy to realistically deform complex shapes. We will use the above defined shape function approximation from the meshless finite element literature to efficiently represent a shape's deformation field. Using this representation, we define energy terms that penalize non-rigid deformations and changes in the shape's volume, while enforcing

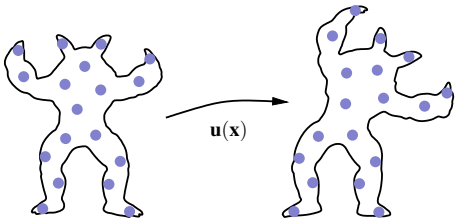

Figure 2: *The goal is to find the deformation field* $\mathbf{u}(\mathbf{x})$ *that deforms the shape in the desired way. We represent this field at discrete nodes* $\mathbf{x}_i$ *and use meshless shape functions to approximate the deformation field over the whole shape as* $\mathbf{u}(\mathbf{x}) = \sum_i \Phi_i(\mathbf{x})\mathbf{u}_i$*, where the displacement vectors* $\mathbf{u}_i$ *are the unknowns we will solve for.*

the user's input constraints. We propose a novel nodal sampling algorithm and define the nodal coupling based material distances to ensure adequate deformation modeling. The resulting deformation framework can be used to interactively model complex shapes. It will also serve as the core component in the deformable keyframe interpolation and motion planning framework that we will discuss in the next section.

3.1. Deformation Field Representation

The object's shape is sampled using a coarse set of nodes (we will discuss our sampling algorithm in Sec. 3.3). The displacements of these nodes will completely define the shape's deformation. Given a set of N nodes $\mathbf{x}_i \in \mathbb{R}^3$ with support radii r_i and deformations $\mathbf{u}_i \in \mathbb{R}^3$, the continuous deformation field is defined using Eq. 1 as (see also Fig. 2)

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x})\mathbf{u}_i. \tag{5}$$

We use a complete linear basis of order $n = 1$ ($\mathbf{p}(\mathbf{x}) = [1\ x\ y\ z]^T$, and $\mathbf{x} = [x\ y\ z]^T$) in the shape function construction. This allows us to reconstruct rigid motions exactly. Eq. 5 maps every point $\mathbf{x}$ in the undeformed shape to its image $\mathbf{x}+\mathbf{u}(\mathbf{x})$ in the deformed shape. For ease of notation we will denote this mapping as

$$\mathbf{f}(\mathbf{x}) = \mathbf{x}+\mathbf{u}(\mathbf{x}). \tag{6}$$

Given the above formulation, our goal is to find the nodal displacement vectors $\mathbf{u}_i$ so that the resulting continuous deformation field $\mathbf{u}(\mathbf{x})$ (or equivalently $\mathbf{f}(\mathbf{x})$) fulfills desirable properties. In the following we will define these goal properties and show how $\mathbf{u}_i$ can be found using an energy minimization procedure. Note first that, because the shape functions are not interpolating, the nodal displacement vectors $\mathbf{u}_i$ are in general not equal to the actual deformations $\mathbf{u}(\mathbf{x}_i)$ evaluated at the node positions ($\mathbf{u}_i \neq \mathbf{u}(\mathbf{x}_i)$). The former are hence often called *fictitious* displacements, while the latter are the *real* nodal displacements.

3.2. Deformation Field Optimization

We wish to find a continuous deformation field $\mathbf{f}(\mathbf{x}) = \mathbf{x}+\mathbf{u}(\mathbf{x})$ that maps the shape to its deformed pose while satisfying following constraints (see also Fig. 3).

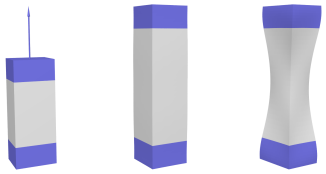

Figure 3: *Illustration of the effect of the different shape modeling constraints. Left: handle constraints are specified to fix the bottom of the box and to move the top to the desired position. Middle: with only the rigidity constraint the total volume is increased by 53%. Right: the total volume remains within 3% of the original when the volume constraint is added.*

Handle Constraints Handle constraints restrict the movement of certain points of the shape. For example, the user may want to fix the legs while pulling one of the arms of the model to deform its shape. Thus, a handle constraint simply states that the deformation field $\mathbf{f}(\mathbf{x}_k)$ should move a given point $\mathbf{x}_k$ to a prescribed target position $\mathbf{x}'_k$. Given a set of K handle constraints $(\mathbf{x}_k, \mathbf{x}'_k)$, we will minimize:

$$E_{handle} = \sum_{k=1}^{K} \|\mathbf{f}(\mathbf{x}_k) - \mathbf{x}'_k\|^2. \tag{7}$$

Rigidity The deformation field is completely rigid if at all points $\nabla\mathbf{f}^T(\mathbf{x})\nabla\mathbf{f}(\mathbf{x}) = \mathbf{I}$, where $\nabla = (\partial/\partial x, \partial/\partial y, \partial/\partial z)$. Hence, to obtain as-rigid-as-possible shape deformations, we will minimize:

$$E_{rigidity} = \int_{\mathbf{x}\in\mathcal{V}} \|\nabla\mathbf{f}^T(\mathbf{x})\nabla\mathbf{f}(\mathbf{x}) - \mathbf{I}\|_F^2 d\mathbf{x}, \tag{8}$$

where the integration is over the (undeformed) shape's volume $\mathcal{V}$ and $\|\cdot\|_F$ is the Frobenius norm. To facilitate optimization, we will only penalize non-rigid behavior at the nodal positions. This leads to the discretized equation

$$E_{rigidity} = \sum_{i=1}^{N} V_i \|\nabla\mathbf{f}^T(\mathbf{x}_i)\nabla\mathbf{f}(\mathbf{x}_i) - \mathbf{I}\|_F^2. \tag{9}$$

Here, $\nabla\mathbf{f}(\mathbf{x}) = \mathbf{I}+\nabla\mathbf{u}(\mathbf{x})$, where $\nabla\mathbf{u}(\mathbf{x})$ is computed using the analytic derivative formula of Eq. 4. The scaling by the node volume $V_i = 4/3\pi r_i^3$ can be omitted when using uniform node radii ($r_i = r$). In the following we will directly write down the discretized equation (cf. Eq. 9) and leave out the continuous one (cf. Eq. 8) for the sake of brevity.

Volume Preservation The deformation field preserves the shape's volume if and only if $\det(\nabla\mathbf{f}(\mathbf{x})) = 1$ over the whole shape. Thus, the deformed shape's volume matches its original volume as closely as possible if we minimize

$$E_{volume} = \sum_{i=1}^{N} V_i(\det(\nabla\mathbf{f}(\mathbf{x}_i)) - 1)^2. \tag{10}$$

The optimal deformation field $\mathbf{f}(\mathbf{x})$ can now be found by minimizing the total sum of constraint energies:

$$E = \lambda_1 E_{handle} + \lambda_2 E_{rigidity} + \lambda_3 E_{volume}, \tag{11}$$

where the parameters λ_1, λ_2 and λ_3 vary the contribution of

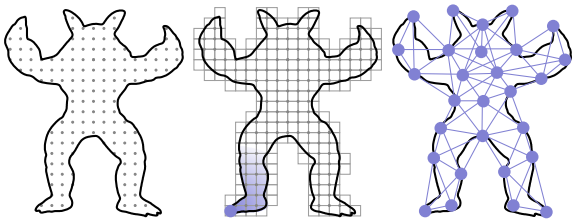

Figure 4: *Topology aware nodal sampling of the shape's interior. Left: the set of candidate points consists of the mesh vertices and a dense set of interior grid points. Middle: nodes are created iteratively by picking the point furthest away from all previously created nodes. A grid-based fast marching method is used to compute distances within the shape. The figure shows the first node and its influence region. Right: the resulting nodal coupling adequately separates parts that are close in Euclidean distance, but far in topological sense such as the legs in the picture.*

each of the different constraints. It can be easily seen from Eq. 7, 9 and 10 that E is a multivariate polynomial of total degree 6 in the unknowns (the fictitious nodal displacements $\mathbf{u}_i$). Minimizing Eq. 11 hence requires a non-linear solver. Note however, that taking analytic derivatives with respect to the unknowns is straightforward.

3.3. Nodal Sampling & Coupling

Given a sampling of the object with N nodes, there are $3N$ unknowns in the deformation optimization procedure. Our goal is to keep this number as low as possible, while allowing realistic shape deformations. When using a coarse set of nodes, it is however important to appropriately define the nodal influence regions (i.e., the non-zero extent of the weight functions $w_i(\mathbf{x})$ in Eq. 2). Simply using Euclidean distances could possibly introduce undesirable artifacts (for example, a node in one leg of a human would incorrectly influence the other leg). In this section we propose a fast nodal sampling algorithm that strategically covers the shape with a low number of nodes while guaranteeing appropriate nodal coupling.

The sampling algorithm creates nodes from a set of candidate points defined as the union of the mesh vertices and a dense set of interior grid points (see Fig. 4, left). It iteratively picks the point $\mathbf{x}_i$ from this set that is furthest away from the already created nodes $\mathbf{x}_{i-1}$, $\mathbf{x}_{i-2}$,..., $\mathbf{x}_0$ until the whole shape is sufficiently covered (the first node is picked randomly). The distance $d(\mathbf{x},\mathbf{x}_i)$ to the node $\mathbf{x}_i$ is computed within the shape by solving the Eikonal equation using a grid-based fast marching method [Set99] (see also the middle image in Fig. 4). This distance represents the *material distance* and corresponds to the length of the shortest path from $\mathbf{x}$ to $\mathbf{x}_i$ without leaving the shape. Using this distance in the weight function $w_i(\mathbf{x})$ to define the nodal shape functions (Eq. 2) ensures that nodes influence the appropriate regions. As noted above, this allows adequate modeling of shape deformations with nearby features such as the fingers of a hand, or the legs of a human.

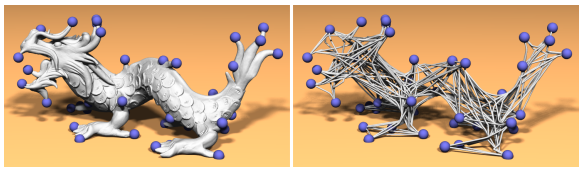

Figure 5: *Nodal sampling for the dragon model. The right image shows the coupling of the nodes as their shortest connecting paths within the dragon. The lengths of these paths are computed using a fast marching algorithm and are used in the shape function computations. Note that this results in a nodal coupling that respects the topology of the object.*

To guarantee non-singular moment matrices (cf. Eq. 3), every point $\mathbf{x}$ in the shape has to be within the support radius of at least 4 non-planar nodes. In the current implementation we prescribe a uniform nodal influence radius $r_i = r$. During node creation we count for each of the original dense candidate points the number of covering nodes, i.e., those nodes $\mathbf{x}_i$ that are within a material distance r_i. The sampling algorithm ends if every point is covered by at least 4 non-planar nodes. The proposed strategy results in a roughly uniform and sufficiently dense nodal sampling.

Note that the fast marching algorithm does not return exact distances. However, it is consistent, in the sense that it converges to the true distances if the grid resolution is increased. Moreover, the computed approximate distances are continuous, which guarantees smooth shape and surface deformations. Because the nodal sampling and coupling is computed in a preprocessing step and does not change during interaction, performance is not a real issue. Typical samplings take in the order of 1 to 5 seconds.

Fig. 5 shows the resulting sampling for the dragon.

3.4. Surface Deformation

Deforming the shape's surface in our framework can be done in a straightforward and very efficient manner. In a preprocessing step, we compute for each mesh vertex $\mathbf{x}$ the set of nodes that have non-zero support at the vertex. Given these nodes, the shape functions $\Phi_i(\mathbf{x})$ and the gradient of the shape functions $\nabla\Phi_i(\mathbf{x})$ are computed using Eq. 1 and Eq. 4 respectively. This computation is only done once before beginning a modeling session. During modeling, the deformed vertex position $\mathbf{x}'$ is computed using Eq. 6 as $\mathbf{x}' = \mathbf{f}(\mathbf{x})$. Note that this simply amounts to computing a linear combination of the neighboring nodes' deformation vectors using the precomputed shape functions. Similarly, the updated (unnormalized) vertex normal $\mathbf{n}'(\mathbf{x})$ can be computed from the local gradient of the deformation field as the matrix-vector product $\mathbf{n}'(\mathbf{x}) = \nabla\mathbf{f}(\mathbf{x})\mathbf{n}(\mathbf{x})$. Here, we can perform a computational trick and reduce the matrix-vector multiplications by scalar-vector multiplications by noting that this expression is equivalent to $\mathbf{n}'(\mathbf{x}) = \mathbf{n}(\mathbf{x}) + \sum_{i=1}^{N}(\nabla\Phi_i^T(\mathbf{x})\mathbf{n}(\mathbf{x}))\mathbf{u}_i$, where the scalars $\nabla\Phi_i^T(\mathbf{x})\mathbf{n}(\mathbf{x})$ are constant and can be precomputed. Again, this amounts to adding to the undeformed

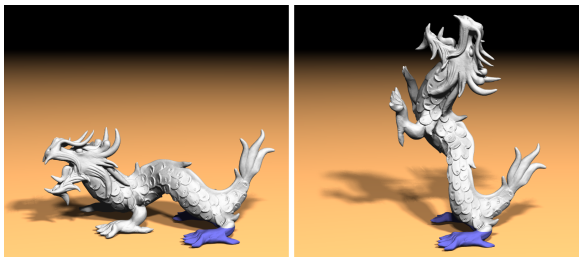

Figure 6: *Deformation of the dragon model obtained using a coarse set of only* 60 *nodes. The nodal deformations are computed on the CPU, while the high resolution surface is deformed faithfully on the GPU. The interaction was performed at a rate of* 55 *fps for the model with* 100*k vertices and* 10 *fps for the model with* 500*k vertices.*

normal a weighted sum of the displacement vectors $\mathbf{u}_i$ where the weights are the precomputed $\nabla\Phi_i^T(\mathbf{x})\mathbf{n}(\mathbf{x})$.

The deformed vertex position and normal can be efficiently computed on the GPU. We store for each vertex $\mathbf{x}$ the indices to the node neighbors and the accompanying scalars $\Phi_i(\mathbf{x})$ and $\nabla\Phi_i^T(\mathbf{x})\mathbf{n}(\mathbf{x})$ in GPU texture memory. During modeling, we only have to send the computed nodal displacement vectors $\mathbf{u}_i$ to the graphics board, which is several orders of magnitude smaller than the number of vertices. Using multiple render passes and fragment shaders we compute and write for each vertex its deformation for the position and normal to intermediate texture memory. Then, in a final render pass, we update the vertex position and normal in a vertex shader using two final texture lookups to retrieve this information.

An example of a deformation of a high resolution dragon model is shown in Fig. 6.

4. Deformable Shape Motions

In this section we extend our framework to allow the computation of smooth deformable shape motions in the presence of obstacles. Using similar techniques as above, we can represent and solve for a time dependent deformation field. Again, by sampling the deformation field only at a sparse discrete number of time instances and by using the meshless shape approximation scheme of Sec. 2, a continuous time dependent deformation field is obtained over the whole time interval. We propose an adaptive temporal sampling strategy that limits the number of unknowns and allows rapid motion path modeling.

4.1. Deformable Motion Field Representation

Each node $\mathbf{x}_i$'s motion path is now sampled at T discrete times t_j, and defined by the fictitious deformation vectors $\mathbf{u}_{i,t_j}$ (see Fig. 7). We call these discrete time representations *frames*. If each frame was treated separately, the deformation at $\mathbf{x}$ would be represented using Eq. 5 as

$$\mathbf{u}_{t_j}(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x})\mathbf{u}_{i,t_j}. \tag{12}$$

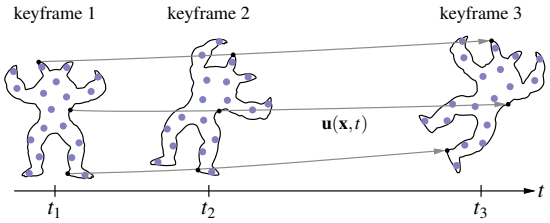

Figure 7: *The goal is to find a smooth motion of the deformable shape that interpolates the keyframes. The continuous time dependent deformation field is defined from the frames as* $\mathbf{u}(\mathbf{x},t) = \sum_{j=1}^{T}\sum_{i=1}^{N}\Phi_j(t)\Phi_i(\mathbf{x})\mathbf{u}_{i,t_j}$ *where the unknowns* $\mathbf{u}_{i,t_j}$ *are the nodal displacements we will solve for. There is one displacement vector* $\mathbf{u}_{i,t_j}$ *for each node* i *in each frame* t_j.

By assigning a support radius r_{t_j} to each frame t_j, we can obtain a smooth time dependent deformation at position $\mathbf{x}$ and time t by approximating over the neighboring frames

$$\mathbf{u}(\mathbf{x},t) = \sum_{j=1}^{T} \Phi_j(t)\mathbf{u}_{t_j}(\mathbf{x}). \tag{13}$$

Here we use one dimensional shape functions ($d = 1$) that guarantee second order consistency ($n = 2$) by using the complete polynomial basis $\mathbf{p}(t) = [1\ t\ t^2]$. This allows us to reconstruct most deformable motions with only a small number of frames. Substituting Eq. 12 in Eq. 13 yields the final expression

$$\begin{aligned}\mathbf{u}(\mathbf{x},t) &= \sum_{j=1}^{T} \Phi_j(t) \sum_{i=1}^{N} \Phi_i(\mathbf{x})\mathbf{u}_{i,t_j} \\ &= \sum_{j=1}^{T}\sum_{i=1}^{N} \Phi_j(t)\Phi_i(\mathbf{x})\mathbf{u}_{i,t_j}.\end{aligned} \tag{14}$$

Hence, the real deformation at position $\mathbf{x}$ at time t is a linear combination of the fictitious deformations $\mathbf{u}_{i,t_j}$ of the spatially neighboring nodes $\mathbf{x}_i$ at neighboring frames. These fictitious deformations are the unknowns we will solve for.

Note that we decouple space and time and use separate shape functions to approximate the displacement field within the shape and to define the continuous displacement field over time. As will be shown below, we fix the spatial sampling, but adapt the temporal sampling iteratively, when solving for the optimal deforming motion. While doing so, we only have to recompute the one dimensional $\Phi_j(t)$, which is cheap, without having to update the nodal sampling and spatial shape functions $\Phi_i(\mathbf{x})$.

4.2. Deformable Motion Field Optimization

Within a frame t_j, we use the rigidity and volume preserving penalties as defined before in Sec. 3.2. To solve for a temporally changing deformation field, we add the following constraints.

Keyframes The user can specify the desired position for the shape at certain keyframe times $t_k, k \in 1\ldots K$. Keyframes are typically defined at the beginning and end of a motion, but can also constrain shape poses at intermediate times. These

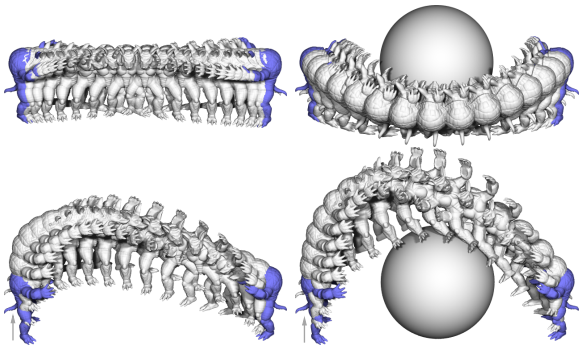

Figure 8: *Illustration of the effect of the different temporal constraints. Top left: smooth interpolation between two keyframes using the keyframe and acceleration constraints. Lower left: result after prescribing the velocity in the first frame (gray arrow). Top right: result after adding an obstacle. Bottom right: result after adding the velocity and obstacle constraints.*

keyframes are converted into handle constraints (cf. Sec. 3.2) by specifying one handle constraint for each node in each keyframe. Note that the user can specify the shape in a keyframe in a *deformed* pose or can only constrain a subset of the shape in a keyframe. We denote the resulting energy function as $E_{keyframe}$ in the following.

Velocity Constraints Along with specifying the shape's keyframe poses, the user can specify a velocity $\mathbf{v}_k$ that the deformation field should satisfy at the keyframes t_k. This velocity should match the temporal derivative of the shape's deformation field at time t_k, for all points in the shape. For all keyframes together and discretized at the nodes, we have to minimize

$$E_{velocity} = \sum_{k=1}^{K} \sum_{i=1}^{N} V_i \| \frac{\partial \mathbf{u}}{\partial t}(\mathbf{x}_i, t_k) - \mathbf{v}_k \|^2. \qquad (15)$$

The analytic time derivatives of the deformation field are computed from the shape function's time derivatives $\partial \Phi_j(t)/\partial t$ using Eq. 4.

Acceleration To obtain smooth motion, we bound the shape's acceleration. Very similar to the above velocity field this yields the energy penalty

$$E_{acceleration} = \sum_{j=1}^{T} \sum_{i=1}^{N} r_{t_j} V_i \| \frac{\partial^2 \mathbf{u}}{\partial t^2}(\mathbf{x}_i, t_j) \|^2, \qquad (16)$$

where r_{t_j} is the support radius of frame t_j. The second derivatives of the displacement field with respect to time can be computed analytically by computing $\partial^2 \Phi_j(t)/\partial t^2$ as detailed in [FM03]. Note again that the resulting expression for the acceleration is a simple linear combination of the fictitious nodal displacements, i.e., the unknowns $\mathbf{u}_{i,t_j}$.

Obstacle Avoidance We define a final penalty function that prevents penetration of the deforming object with possible obstacles in the scene. We assume that the obstacles can be represented by a (time dependent) distance field $d(\mathbf{x},t)$ and that a point $\mathbf{x}$ is penetrating at time t if $d(\mathbf{x},t) \geq 0$. Using this distance field representation, we obtain the following energy penalty function where the outer integration is over the whole time interval and $\mathbf{f}(\mathbf{x},t) = \mathbf{x} + \mathbf{u}(\mathbf{x},t)$.

$$E_{obstacles} = \sum_{j=1}^{T} \sum_{i=1}^{N} r_{t_j} V_i d^2(\mathbf{f}(\mathbf{x}_i, t_j), t_j). \qquad (17)$$

Note that although this energy penalizes collisions for the nodes at the frames, it does not prevent collisions of all points at all times. To prevent artifacts rising from only constraining the nodes, we fatten the nodes to spheres (by using their support radii r_i) and constrain these spheres to be outside the obstacles (hence, we use $d(\mathbf{f}(\mathbf{x}_i,t_j),t_j) + r_i$ instead of $d(\mathbf{f}(\mathbf{x}_i,t_j),t_j)$ in the above equation). If the union of these spheres covers the whole shape, this adequately prevents penetrations at the frames t_j. However, nothing restricts the shape from colliding with obstacles at other time instances, as the interpolation scheme is collision oblivious. To deal with this issue, Sec. 4.3 presents an adaptive time discretization scheme that iteratively adds new frames t_j where these problems occur.

Given the discrete spatial and temporal sampling, the optimal time dependent deformation field $\mathbf{f}(\mathbf{x},t) = \mathbf{x} + \mathbf{u}(\mathbf{x},t)$ can now be found by minimizing the total energy

$$\begin{aligned} E \;=\; & \lambda_1 E_{keyframe} + \lambda_2 E_{rigidity} + \lambda_3 E_{volume} \\ & \lambda_4 E_{velocity} + \lambda_5 E_{acceleration} + \lambda_6 E_{obstacles}, \end{aligned} \qquad (18)$$

where λ_1 to λ_6 are again parameters to modify the contribution of the various constraints. Similar to the energy function of Eq. 11, the above equation is a polynomial of total degree 6 in the unknown nodal displacements $\mathbf{u}_{i,t_j}$.

We now discuss how the deformation field is temporally discretized by iteratively creating and solving for new frames.

4.3. Adaptive Temporal Sampling

In the (single frame) deformation modeling part of Sec. 3, the total number of unknowns to solve for is $3N$, where N is the number of nodes. In the motion planning setting, the total number of unknowns multiplies to $3NT$, where T is the number of frames. To keep this number sufficiently low, we propose an adaptive time sampling strategy that introduces frames iteratively in problematic regions (see Fig. 9).

Initially, we only have frames t_j that correspond to the keyframes specified by the user (see top left image in Fig. 9). We optimize the displacement field as discussed above and evaluate the error at a dense number of frames in between the frames t_j (we typically evaluate at 10 intermediate frames). We then introduce a new frame t_j at the time instance t_{max} where the error is maximal. We solve again and iterate until a desired accuracy is obtained. When we introduce a new frame at time t_{max}, we initialize the nodes' deformation vectors at the new frame as $\mathbf{u}_{i,t_{max}} = \sum_{t=1}^{T} \Phi_j(t_{max}) \mathbf{u}_{i,t_j}$. This yields a good initial guess for the subsequent solve.

The proposed adaptive sampling strategy greatly reduces the

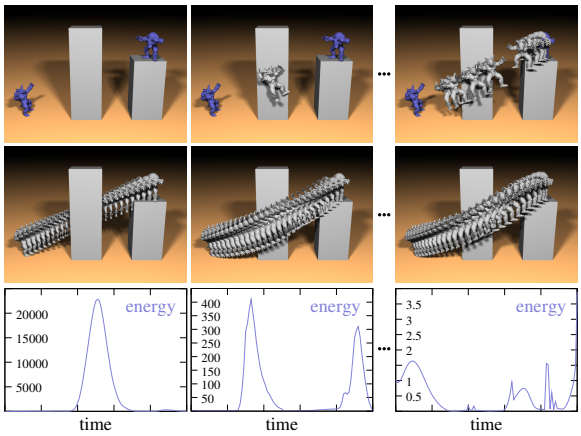

Figure 9: *The user wants to find a smooth motion between two deformed shapes of the armadillo, that are specified as keyframes in the top left image. The linearly interpolated motion between these two frames is inadequate and has high energy where the armadillo moves through one of the obstacles (left column). Our algorithm adds a new frame and solves for the armadillo's deformed shape at this time step. The resulting motion largely avoids the obstacles (middle column). This process is iterated until a sufficiently low energy is obtained for 10 frames (right column).*

number of unknowns and introduces frames only at problematic regions, for example when there is high acceleration, or when the deforming shape is penetrating an obstacle (see for example Fig. 9). Thanks to the second order consistency in the temporal shape functions $\Phi_j(t)$, we can represent most motions by only a very low number of frames. The frames' support radii r_{t_j} (and hence weight functions $w_{t_j}(t)$) are adapted to the frame spacing to ensure that every time instance t is covered by at least 3 neighboring frames. This guarantees non-singular moment matrices and safe computation of the temporal shape functions. If only two frames are present (as in the beginning of Fig. 9), we use simple linear interpolation to define the shape's deformation field.

5. Numerical Solver

In addition to reducing the number of unknowns, our nonlinear energy function formulation has a few additional advantages. Most importantly, it makes it easy to compute the gradient of the objective function analytically because $\mathbf{u}(\mathbf{x},t)$ and all of its derivatives are linear combinations of the unknowns. Moreover, the objective function is a low-degree polynomial in the unknowns, and thus standard optimization techniques perform well in the vicinity of the solution. In the deformation tool, we use the current positions of the nodes as the initial guess for the new deformation, during manipulations of the object. This ensures a good initialization because instantaneous deformations are small and we expect a stable deformation field. During multiframe deformable motion field optimization, we use the aforementioned meshless function interpolation to initialize deformations for intermediate frames. This means that if every individual point follows a trajectory that is locally well approximated by a quadratic, our initial guess will be very close to the optimal solution. In practice, we use the non-linear LBFGS solver from OPT++. As mentioned above, our energy function may have several local minima. In order to control the convergence of the optimizer to the desired solution, we implemented an interactive tool that allows to manually control the initial guess for the deformation at any intermediate frame. This is particularly useful during adaptive temporal sampling, because it allows the user to control the general direction of the motion without specifying handle constraints. Our video demonstrates one application of this technique.

vertices \ nodes	20	50	100	200
100k	4.6/4.4	9.5/4.6	15.2/4.7	31.6/5.5
250k	4.9/51.5	9.6/52.2	14.5/51.8	33.0/55.2
500k	5.1/95.9	9.4/94.8	15.6/97.6	32.5/102.9

Table 1: *Timing statistics (ms) for the dragon deformation of Fig. 6 for different numbers of vertices and nodes. Each entry shows the average time spent solving on the CPU and on deforming and rendering the triangle mesh on the GPU.*

6. Results & Discussion

We implemented our algorithms in C++ and used Cg for the fragment and vertex shaders that compute the deformed mesh vertices. Our models are given as triangle meshes. In the preprocessing step we compute a regular distance field and use the same grid for the nodal sampling and the fast marching. Inversion of the moment matrices is performed using the Cholesky decomposition code of the JAMA/C++ linear algebra package. The interactive results in the accompanying video are rendered using OpenGL and Cg. The results in the paper are visualized using POV-Ray. All results in this paper are obtained on a 3.2 GHz Intel Pentium D CPU with an NVIDIA GeForce 8800 graphics board.

Fig. 6 shows the result of a real-time deformation of the dragon model. We obtain an interaction rate of 10 fps for the model of 500k vertices and at least 55 fps for the decimated model of 100k vertices, both sampled with 60 nodes (see also the accompanying video). The parameter settings in Eq. 11 for this example are $\lambda_1 = 10000$, $\lambda_2 = 3$, $\lambda_3 = 10$. Detailed timings for varying numbers of vertices and nodes are given in Table 1. The slow-down for the 250k and 500k models is due to a rendering bottleneck. Note that even for the 500k model, we obtain similar deformation results as [BPWG07], but at almost two orders of magnitude faster interaction rate. This is mainly thanks to the much lower number of necessary deformation nodes in our approach and the very efficient surface deformation algorithm.

Fig. 9 illustrates the adaptive temporal sampling algorithm. Starting with only the keyframes, a new frame is iteratively added where the energy is the largest until a sufficiently accurate motion is obtained with 10 frames. We used 53 nodes to sample the armadillo, resulting in a total of $3 \times 10 \times 53 = 1590$ scalar unknowns to solve for. The total solve time took 6 seconds. The different energies in Eq. 18 are weighted by

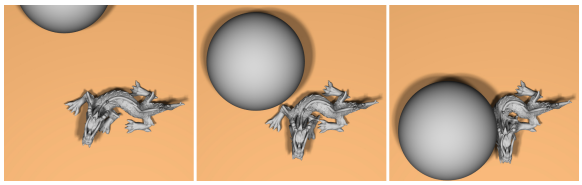

Figure 10: *The dragon deforms to avoid the moving obstacle. Note how the dragon anticipates the impact of the ball thanks to the acceleration constraint.*

$\lambda_1 = 10000$, $\lambda_2 = 1$, $\lambda_3 = 10$, $\lambda_4 = 10$, $\lambda_5 = 0.01$, $\lambda_6 = 1000$. We use the same settings for all motion interpolation examples. In the accompanying movie we also show how the user can modify the result to obtain a solution corresponding to a different local minimum of the objective function.

Fig. 10 shows the resulting motion of a deforming dragon in the presence of a moving obstacle. The initial and final keyframe are set to the undeformed dragon and its back feet are fixed over the whole time interval. The intermediate motion is computed automatically. Note how the dragon anticipates the moving obstacle and tries to avoid it by bending away due to our acceleration penalty. This motion took 10 seconds to compute with 59 nodes and 25 frames introduced by the adaptive temporal sampler, where 17 of the 25 frames are concentrated in the middle of the animation.

Fig. 1 shows a long animation sequence obtained by specifying 7 keyframe poses of the armadillo. The armadillo is sampled by 66 nodes, its triangle mesh contains almost 166k vertices. The keyframe poses were modeled in 2.5 minutes during which the interaction rate was 60 fps. The number of frames used for the motion computation was 28 and the solve time took 16 seconds. To obtain realistic bouncing behavior, we allow C^1 discontinuities in the resulting motion path. We achieve this by omitting the acceleration constraint at the third and fifth keyframe and by restricting the support of the temporal shape functions to the intervals defined by the first and third, third and fifth, and the fifth and last keyframe respectively. Note how the armadillo deforms realistically, even if its shape in two consecutive keyframes differs significantly.

Finally, we compare our adaptive temporal sampling strategy with uniform sampling for a simple test case involving rigid motions (see Fig. 11). As can be seen in Table 2, many more frames are needed when the frames are uniformly spaced over the time interval. The adaptive sampler will however strategically introduce the bulk of the frames at the end of the animation where the armadillo has to make a rapid 180 degree rotation and use less frames where the armadillo performs a simple translation. We obtained similar speedups for deforming objects. For example, for the result of Fig. 10, a regular time interval sampling would require 65 frames and a total computation time of 27 seconds to converge with the same total energy as for the adaptive sampler.

Our method allows easy and intuitive modeling by just selecting and dragging points on the mesh and by simply specifying a few keyframe poses. We avoid the problem of manually designing an appropriate control cage or of computing good quality tetrahedralizations in classical finite element methods. Our sampling algorithm automatically computes an adequate set of nodes, the only required input is the shape's boundary mesh. Nodes are spread uniformly over the shape's material and the shape functions adapt automatically to the sample spacing and the input mesh. Our temporal sampling algorithm automatically introduces frames in critical regions while keeping the number of unknowns low. We showed that using this coarse representation complex shapes and motions can be modeled efficiently.

Figure 11: *Test scene (keyframes from left to right) for the adaptive and regular temporal sampling comparisons given in Table 2.*

	frames	energy	time
adaptive	33	0.8	6s
uniform	29	8.2	7s
uniform	78	0.8	19s
uniform	113	0.4	22s

Table 2: *Sampling strategy comparisons for Fig. 11.*

The proposed technique has a few limitations. First, our temporal deformation approximation scheme does not reconstruct rotations exactly. Indeed, it is only second order accurate in time. However, most motions are locally quadratic and can therefore be represented using only a few basis frames. As future work we plan to design temporal basis functions that not only reconstruct translations exactly but also rotations. Second, our sampling algorithm relies on a proper inside/outside classification of the shape. However, the meshless representation and optimization procedure does not require a closed shape and other sampling algorithms (e.g., [WAO*08]) could potentially be used to obtain suitable nodal samplings for meshes with boundaries. Finally, we wish to extend our method to allow skeleton based deformations. By defining handle constraints for skeleton bones, it should be possible to create articulated deformations.

Acknowledgments The authors wish to acknowledge the support of NSF grants ITR 0205671 and FRG 0354543, NIH grant GM-072970, DARPA grant HR0011-05-1- 0007, and the Max-Planck Center for Visual Computing and Communication. Bart Adams is funded as a post-doctoral researcher by the Fund for Scientific Research, Flanders (F.W.O.-Vlaanderen). We would like to thank the Stanford University Computer Graphics Laboratory for providing the armadillo model and XYZ RGB Inc. for providing the dragon model.

References

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-rigid-as-possible shape interpolation. In *Proceedings of ACM SIGGRAPH 2000* (2000), ACM Press/Addison-Wesley Publishing Co.

[AFTCO07] AU O. K.-C., FU H., TAI C.-L., COHEN-OR D.: Handle-aware isolines for scalable shape editing. *ACM Trans. Graph. 26*, 3 (2007).

[BK03] BOTSCH M., KOBBELT L.: Multiresolution surface representation based on displacement volumes. *Computer Graphics Forum 22*, 3 (2003).

[BPGK06] BOTSCH M., PAULY M., GROSS M., KOBBELT L.: Primo: coupled prisms for intuitive surface modeling. In *Proceedings of the fourth Eurographics symposium on geometry processing* (2006), Eurographics Association.

[BPWG07] BOTSCH M., PAULY M., WICKE M., GROSS M.: Adaptive space deformations based on rigid cells. *Computer Graphics Forum 26*, 3 (2007).

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (2008).

[BSS07] BOUBEKEUR T., SORKINE O., SCHLICK C.: Simod: Making freeform deformation size-insensitive. In *IEEE/Eurographics Symposium on Point-Based Graphics 2007* (2007).

[DSP06] DER K. G., SUMNER R. W., POPOVIĆ J.: Inverse kinematics for reduced deformable models. *ACM Trans. Graph. 25*, 3 (2006).

[FKR05] FLOATER M. S., KÓS G., REIMERS M.: Mean value coordinates in 3d. *Comput. Aided Geom. Des. 22*, 7 (2005).

[FM03] FRIES T.-P., MATTHIES H. G.: *Classification and Overview of Meshfree Methods*. Tech. rep., TU Brunswick, Germany Nr. 2003-03, 2003.

[GP07] GROSS M., PFISTER H.: *Point-Based Graphics (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann Publishers Inc., 2007.

[GQ05] GUO X., QIN H.: Real-time meshless deformation: Collision detection and deformable objects. *Comput. Animat. Virtual Worlds 16*, 3-4 (2005).

[Hof04] HOFER M.: *Variational motion design in the presence of obstacles*. PhD thesis, Vienna University of Technology, 2004.

[HPR04] HOFER M., POTTMANN H., RAVANI B.: From curve design algorithms to the design of rigid body motions. *Vis. Comput. 20*, 5 (2004).

[HSL*06] HUANG J., SHI X., LIU X., ZHOU K., WEI L.-Y., TENG S.-H., BAO H., GUO B., SHUM H.-Y.: Subspace gradient domain mesh deformation. *ACM Trans. Graph. 25*, 3 (2006).

[JMD*07] JOSHI P., MEYER M., DEROSE T., GREEN B., SANOCKI T.: Harmonic coordinates for character articulation. *ACM Trans. Graph. 26*, 3 (2007).

[JP02] JAMES D. L., PAI D. K.: Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. *ACM Trans. Graph. 21*, 3 (2002).

[JT05] JAMES D. L., TWIGG C. D.: Skinning mesh animations. *ACM Trans. Graph. 24*, 3 (2005).

[KKiA05] KONDO R., KANAI T., ICHI ANJYO K.: Directable animation of elastic objects. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on computer animation* (2005), ACM.

[KMP07] KILIAN M., MITRA N. J., POTTMANN H.: Geometric modeling in shape space. *ACM Trans. Graph. 26*, 3 (2007).

[LKCOL07] LIPMAN Y., KOPF J., COHEN-OR D., LEVIN D.: Gpu-assisted positive mean value coordinates for mesh deformations. In *Proceedings of the fifth Eurographics symposium on geometry processing* (2007), Eurographics Association.

[LSCO*04] LIPMAN Y., SORKINE O., COHEN-OR D., LEVIN D., RÖSSL C., SEIDEL H.-P.: Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International* (2004), IEEE Computer Society Press.

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point based animation of elastic, plastic and melting objects. *Proceedings of 2004 ACM SIGGRAPH Symposium on Computer Animation* (2004).

[NMK*05] NEALEN A., MÜLLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically Based Deformable Models in Computer Graphics. In *Eurographics: State of the Art Report* (2005).

[OFTB96] ORGAN D., FLEMING M., TERRY T., BELYTSCHKO T.: Continuous meshless approximations for nonconvex bodies by diffraction and transparency. *Comput. Mechanics 18* (1996).

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTRÉ P., GROSS M., GUIBAS L. J.: Meshless animation of fracturing solids. *ACM Trans. Graph. 24*, 3 (2005).

[PSE*00] POPOVIĆ J., SEITZ S. M., ERDMANN M., POPOVIĆ Z., WITKIN A.: Interactive manipulation of rigid body simulations. In *Proceedings of ACM SIGGRAPH 2000* (2000), ACM Press/Addison-Wesley Publishing Co.

[PSE03] POPOVIĆ J., SEITZ S. M., ERDMANN M.: Motion sketching for control of rigid-body simulations. *ACM Trans. Graph. 22*, 4 (2003).

[Set99] SETHIAN J. A.: *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[SK04] SHEFFER A., KRAEVOY V.: Pyramid coordinates for morphing and deformation. In *Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium* (2004), IEEE Computer Society.

[SLCO*04] SORKINE O., LIPMAN Y., COHEN-OR D., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2004), Eurographics Association.

[SSP07] SUMNER R. W., SCHMID J., PAULY M.: Embedded deformation for shape manipulation. *ACM Trans. Graph. 26*, 3 (2007).

[Sto07] STOLL C.: *A Volumetric Approach to Interactive Shape Editing*. Research Report MPI-I-2007-4-004, Max-Planck-Institut für Informatik, June 2007.

[SYBF06] SHI L., YU Y., BELL N., FENG W.-W.: A fast multigrid algorithm for mesh deformation. *ACM Trans. Graph. 25*, 3 (2006).

[SZGP05] SUMNER R. W., ZWICKER M., GOTSMAN C., POPOVIĆ J.: Mesh-based inverse kinematics. *ACM Trans. Graph. 24*, 3 (2005).

[SZT*07] SHI X., ZHOU K., TONG Y., DESBRUN M., BAO H., GUO B.: Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph. 26*, 3 (2007).

[vFTS06] VON FUNCK W., THEISEL H., SEIDEL H.-P.: Vector field based shape deformations. *ACM Trans. Graph. 25*, 3 (2006).

[WAO*08] WAND M., ADAMS B., OVSJANIKOV M., BERNER A., BOKELOH M., JENKE P., GUIBAS L., SEIDEL H.-P., SCHILLING A.: Efficient reconstruction of non-rigid shape and motion from real-time 3d scanner data. *submitted to ACM Trans. Graph.* (2008).

[WP95] WITKIN A., POPOVIC Z.: Motion warping. In *Proceedings of ACM SIGGRAPH 95* (1995), ACM.

[XZWB05] XU D., ZHANG H., WANG Q., BAO H.: Poisson shape interpolation. In *Proceedings of the 2005 ACM symposium on solid and physical modeling* (2005), ACM.

[XZY*07] XU W., ZHOU K., YU Y., TAN Q., PENG Q., GUO B.: Gradient domain editing of deforming mesh sequences. *ACM Trans. Graph. 26*, 3 (2007).